When even the interface evolves...

Alexandre Madeira
HASLab - INESC TEC
Dep. Mathematics, U. Aveiro
Critical Software S.A.
Email: madeira@ua.pt

Renato Neves
HASLab - INESC TEC
Univ. Minho
Email:nevrenato@gmail.com

Manuel A. Martins
CIDMA
Dep. Mathematics, U. Aveiro
Email: martins@ua.pt

Luis S. Barbosa
HASLab - INESC TEC
Univ. Minho
Email: lsb@di.uminho.pt

***Abstract*—This paper extends the authors' previous work on a formal approach to the specification of reconfigurable systems, introduced in [7], in which configurations are taken as local states in a suitable transition structure. The novelty is the explicit consideration that not only the realisation of a service may change from a configuration to another, but also the set of services provided and even their functionality, may themselves vary. In other words, interfaces may evolve, as well.**

I. Introduction

Context: configurations-as-local-states

The qualifier *reconfigurable* is used for systems whose execution modes, and not only the values stored in their internal memory, may change in response to the continuous interaction with the environment. In [7] the authors' introduced a formal, two level approach to their specification. The *rationale* sought to combine two basic dimensions in systems specification: one which emphasizes *behaviour* and its evolution, another focused on *data* and their transformations.

Behaviour is typically specified through (some variant of) *state-machines*. Such models capture evolution in terms of event occurrences and their impact in the system's internal state configuration (see e.g. [6]). Data types and services upon them, on the other hand, are often presented as theories in suitable logics, over a signature which offers a syntactic interface to the system. Semantics is, then, given by a class of concrete algebras acting as models for the specified theory (see e.g. [9]). The starting point for our approach, is that these dimensions are interconnected: the functionality offered by a reconfigurable system, at each moment, may depend on the stage of its evolution. In [7] the reconfiguration dynamics is modelled as a transition system, whose nodes are interpreted as the different configurations it may assume. Therefore, each of such nodes is endowed with an algebra, or even a first-order structure, to formally characterise the semantics of the services offered in the corresponding configuration. Technically, models of reconfigurable systems are given as *structured* state-machines whose states denote *algebras*, rather than *sets*.

The approach assumed, however, that the signature, *i.e.*, the *interface* provided at any local state is fixed. Or, to put this in an equivalent way, that the system's interface is invariant with respect to the reconfiguration process. This paper, as explained below, aims at relaxing this condition.

Before that, however, a word is due on the specification logic adopted. Modal languages are, quite obviously, the natural choice to talk about transition systems. We resort, however, to their *hybrid* extension [1] because a crucial point in the whole approach is to be able to express and verify properties which may only hold in a specific state, or a group thereof. Hybrid logic introduces a special set of symbols to name states and a suitable family of reference operators. Additionally, at each state, a whole algebra has to be specified. This entails the combination of hybrid and modal features with equational logic, leading to adoption of a variant of *hybrid equational logic*. This combination, in a highly general setting, is discussed in a complementary line of research, documented in [8], [4], but is not essential for what follows. The overall approach, summed up in the slogan *configurations-as-local-states*, is sketched in Fig I.

It should be stressed this approach differs from the one proposed by Y. Gurevich in the early nineties under the designation of *evolving algebras* and, later, as *abstract state machines* [5], [2]. A state there encapsulates a specific configuration of variables in an algebra: as configurations change, so the algebra *evolves*. In our own approach, however, each node corresponds to a different, independent algebra.

Contribution: reconfigurable interfaces

As mentioned above, we intend to go a step further and allow not only a possibly different algebra in each state, but also different algebras over a different signature. Actually, in a number of cases the services a system may offer, and their functionalities, may depend on the particular configuration or mode of operation the system is currently assuming. Therefore, in the place of a unique (static) interface (S, F), we consider in the sequel a family of signatures $\left(S^i, F^i\right)_{i\in \mathrm{Nom}}$, indexed by the set Nom of state identifiers, *i.e.*, the nominals of our hybrid specification language. The approach introduced in [7] is extended accordingly. Technically, this is achieved through the introduction of (hybrid) *partial algebra*-specifications to "simulate" the intended, independent (hybrid) *equational* ones. Note, however, that, even resorting to *partial* specifications, models will always be (total) algebras with respect to the corresponding local interface. The following example will be used to illustrate the method.

Example. Suppose that, in the context of a client server architecture, a buffering component is required to store and manage incoming messages from different clients. Depending on the server's execution mode, *i.e.*, on its current configuration, issues like the order in which calls have arrived or the number of repeated messages may, or may not, be relevant. Therefore, the *shape* of the buffering component may vary, typically being determined by an external manager.

A model for this component comprises four endowed,

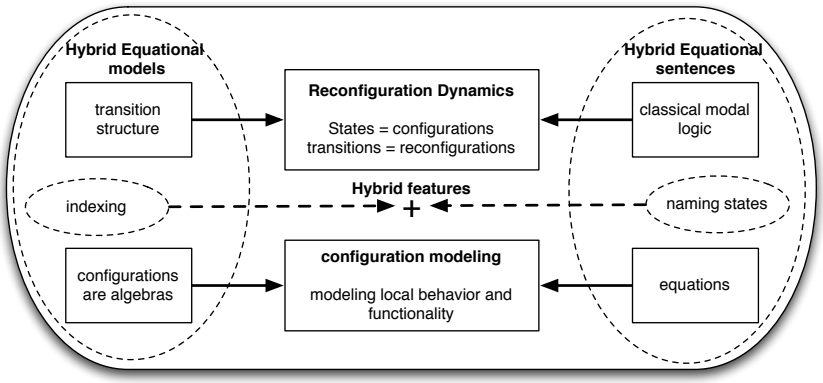

Fig. 1. The specification approach

respectively with i) an algebra of *sequences* (for configurations where both order and multiplicities are relevant issues), ii) an algebra of *multi sets* (when the order may be left out), iii) an algebra of *sets* (when the application may abstract over order and repetitions), and finally iv) an algebra of *repetition free sequences* (to cater only the messages' order). Going from one configuration to another involves not only a change in the way a service is realised (*e.g.*, insertion clearly differs from one state to the other), but also a change at the *interface* level. For example, an operation to count the number of replicated messages does not make sense if sets are used as a local model.

Outline. The remaining of the paper is organized as follows: Section II recalls the definitions of *equational* and *partial algebra* logics as well as the notions of *hybridization* and *presentation of logics*. The main contribution appears in Section III. Finally, Section IV briefly discusses tool support for the extended specification method.

II. Background

The Equational Logic EQ

Services and system operations are specified in equational logic. A signature is a pair (S, F) where S is a set of sort symbols and $F = \{F_{\underline{ar}\to s} \mid \underline{ar} \in S^*, s \in S\}$ is a family of sets of operation symbols indexed by arities $\underline{ar}$ (for the arguments) and sorts s (for the results). A *model* M for (S, F) is an (S, F)-*algebra* which interprets each sort symbol s as a set M_s, its carrier, and each operation symbol $\sigma \in F_{\underline{ar}} \to s$ as a function $M_\sigma : M_{\underline{ar}} \to M_s$, where $M_{\underline{ar}}$ is the product of the carriers of sorts in $\underline{ar}$. The *sentences* are the universal quantified (S, F)-equations $(\forall X)t = t'$. The satisfaction relation is defined recursively on the structure of the sentences:

- $M \models^{EQ}_{(S,F)} t = t'$ iff $M_t = M_{t'}$, where M_t denotes the interpretation of term t in M defined recursively by $M_{\sigma(t_1,\ldots,t_n)} = M_\sigma(M_{t_1}, \ldots, M_{t_n})$.
- $M \models^{EQ}_{(S,F)} (\forall X)\rho$ iff $M' \models^{EQ}_{(S,F+X)} \rho$, for any $(S, F+X)$-expansion M' of M.

The Partial Algebras Logic PA

In this case, signatures are tuples (S, TF, PF), where TF and PF are families of sets of, respectively, *total* and *partial* function symbols such that $TF_{\underline{ar}\to s} \cap PF_{\underline{ar}\to s} = \emptyset$ for each arity $\underline{ar}$ and each sort s. Models of PA are partial algebras, *i.e.* function symbols are interpreted as partial, rather than total, functions. The *sentences* are defined by the following grammar

$$\rho, \rho' \ni t \stackrel{s}{=} t' \mid t \stackrel{e}{=} t' \mid \mathsf{df}(t) \mid \neg\rho \mid \rho \odot \rho'$$

where $\odot \in \{\wedge, \vee, \Rightarrow\}$. Finally, the *satisfaction relation* is

- $A \models^{PA}_{(S,TF,PT)} \mathsf{df}(t)$ iff A_t is defined;
- $A \models^{PA}_{(S,TF,PT)} t \stackrel{s}{=} t'$ iff $A_t = A_{t'}$ when both are defined;
- $A \models^{PA}_{(S,TF,PT)} t \stackrel{e}{=} t'$ iff A_t and $A_{t'}$ are defined and $A_t = A_{t'}$

with the usual interpretation of the boolean connectives.

Hybridization

The combination of both equational and partial algebras logics with a hybrid language plays a crucial role in the specification method discussed in this paper. Both are briefly presented below. Following this description it is not hard to notice the emergence of a common pattern in the way a hybrid logic is built on top of another logic. Such a construction was made systematic as part of this research effort, in [8], [4]. The so-called *hybridization* process is formulated in the general setting of the theory of institutions [3].

Hybrid equational logic $\mathcal{H}EQ$ is presented as follows: signatures are triples $\Delta = (\Sigma, \mathrm{Nom}, \Lambda)$, where Σ is an equational signature (S, F), Nom and Λ are sets of symbols for nominals and modalities, respectively. The sentences are defined by the following grammar

$$\rho, \rho' \ni i \mid \rho_0 \mid @_i\rho \mid \rho \odot \rho' \mid \neg\rho \mid [\lambda]\rho$$

where ρ_0 is an equational sentence over Σ, $i \in \mathrm{Nom}$ and $\odot \in \{\wedge, \vee, \Rightarrow\}$. As usual $\langle\lambda\rangle\rho$ denotes $\neg[\lambda]\neg\rho$. Models are (Nom, Λ)-transition structures with a Σ-algebra associated to each state. Formally, models of Δ are pairs (M, W) where

- $|W|$ is a set, for each $\lambda \in \Lambda$, $W_\lambda \subseteq |W| \times |W|$ is a binary relation and for each $i \in \mathrm{Nom}$, W_i is a constant in $|W|$;

- $M : W \to \mathrm{Mod}^{EQ(S,F)}$ is a function associating algebras to states. $M(w)$ is denoted by M_w.

Finally, the satisfaction relation is defined as follows:

- $(M, W) \models^w_\Delta i$ iff $W_i = w$
- $(M, W) \models^w_\Delta \rho_0$ iff $M_w \models^{EQ}_\Sigma \rho_0$
- $(M, W) \models^w_\Delta @_i\rho$ iff $(M, W) \models^{W_i}_\Delta \rho$
- $(M, W) \models^w_\Delta \rho \wedge \rho'$ iff $(M, W) \models^w_\Delta \rho$ and $(M, W) \models^w_\Delta \rho'$; and analogously for the remaining boolean connectives.

The Hybrid Partial Algebra logic $\mathcal{H}PA$ is defined as $\mathcal{H}EQ$ but taking, a partial algebra signature (S, TF, PF) in place of Σ; a (S, TF, PF)-sentence of PA in place of ρ_0; a function $M : |W| \to \mathrm{Mod}^{PA}(S, TP, PF)$ in place of $M : |W| \to \mathrm{Mod}^{EQ}(S, F)$; and $(M, W) \models^w_\Delta \rho_0$ iff $M_w \models^{PA}_\Sigma \rho_0$ in place of $M_w \models^{EQ}_\Sigma \rho_0$.

Presentations

A common way to define a new logic is to take another logic, plus some additional data suitably expressed through new axioms. One way to proceed is through *presentations*. Formally, given a logic $\mathcal{I}$, a presentation of $\mathcal{I}$, say $\mathcal{I}^{\mathrm{pres}}$, takes as signatures pairs (Σ, Γ) where Σ is a $\mathcal{I}$-signature and Γ a set of $\mathcal{I}$-sentences. The (Σ, Γ)-models of $\mathcal{I}^{\mathrm{pres}}$, are $\mathcal{I}$-models such that $M \models^{\mathcal{I}}_\Sigma \Gamma$. The satisfaction relation of $\mathcal{I}^{\mathrm{pres}}$ is the restriction of $\mathcal{I}$ to $\mathcal{I}^{\mathrm{pres}}$-models.

III. Reconfiguration of interfaces

The *hybridization* process mentioned above assumes that all configurations have models over the same signature. Suppose, however, that this is not the case and one has instead a family of different equational signatures $(S^i, F^i)_{i\in \mathrm{Nom}}$, one for each (named by nominal i) state. A global signature can be obtained in $\mathcal{H}PA$ as follows. The first step is to define a signature (S, TF, PF) in PA able to capture the all possible possible interfaces in $(S^i, F^i)_{i\in \mathrm{Nom}}$. Thus, operations are split into the ones which are globally defined (*i.e.*, present in any (S^i, F^i), for $i \in \mathrm{Nom}$) and on those which concern only a specific state named, say, by $i \in \mathrm{Nom}$. These two sets of operations defines a (global) PA-signature (S, TF, PF) containing all of these operations as follows

$$S = \bigcup_{i\in \mathrm{Nom}} Si$$
$$TF_{\underline{\mathrm{ar}}\to w} = \{\sigma | \sigma \in \bigcap_{i\in \mathrm{Nom}} F^i\}$$
$$PF_{\underline{\mathrm{ar}}\to w} = \{\sigma | \sigma \in (F^i)_{\underline{\mathrm{ar}}\to w} \setminus TF_{\underline{\mathrm{ar}}\to w}, i \in \mathrm{Nom}\}$$

Now, considering the set reconfiguration events Λ which trigger reconfigurations and give rise to the relevant modalities we achieve ar the $\mathcal{H}PA$ signature $((S, TF, PF), \mathrm{Nom}, \Lambda)$. However, the information about which of those operations are defined in each configuration has to be considered. This is done by the following axioms:

$$\Gamma = \begin{array}{l} \{@_i(\forall X)\mathsf{df}(\sigma(X)) | \sigma \in (F^i)_{\underline{\mathrm{ar}}\to w} \cap PF_{\underline{\mathrm{ar}}\to w}, i \in \mathrm{Nom}\} \cup \\ \{\neg @_i(\exists X)\mathsf{df}(\sigma(X)) | \sigma \in PF_{\underline{\mathrm{ar}}\to w} \setminus (F^i)_{\underline{\mathrm{ar}}\to w}, i \in \mathrm{Nom}\} \end{array}$$

Hence, the presentation $\big(((S, TF, PF), \mathrm{Nom}, \Lambda), \Gamma\big)$ in $\mathcal{H}EQ^{\mathrm{pres}}$ contains the intended information. With this signature the method in [7] can be safely applied from this point on. In broad terms, we are going to simulate *local, total* functions with *global, partial* ones. This entails the need for adopting strong equality to specify "global properties" of operations defined in a specific configuration. For instance, any existencial equation $t \overset{\mathrm{e}}{=} t'$ involving operations in PF is inconsistent because it fails on the configurations where these operations are not defined. Of course, this is not the case of existencial equations prefixed by satisfaction operators, i.e., of sentences of form $@_i(t \overset{\mathrm{e}}{=} t')$. But, in general, this is not enough: all operations must be "locally"-total or "completely"-undefined.

In this context, specifications are built according to the following steps:

1) Enumerate the set of relevant configurations and define the set of their names Nom accordingly;
2) Enumerate the set of reconfiguration-events and define for each of them a modality in Λ;
3) Collect the family of local, Nom-indexed interfaces and define $(S_i, F_i)_{i\in \mathrm{Nom}}$;
4) Take the global signature $\big(((S, TF, PF), \mathrm{Nom}, \Lambda), \Gamma\big)$ in $\mathcal{H}EQ^{\mathrm{pres}}$ through the suggested construction;
5) Develop the specification along the lines proposed in [7]:
   a) Specify the global properties, i.e., properties holding in all the system configurations (using strong equations);
   b) Specify the local properties using the satisfaction operators $@_i$ tagged (S_i, F_i)-equations;
   c) Specify the reconfiguration structure, *i.e.*, the underlying transition system using the available modalities.

The method can be illustrated with the example introduced in Section I. First of all define a set $\mathrm{Nom} = \{OM, Om, oM, om\}$ of nominals, where the capitalized letters correspond to the relevance of *order* and *multiplicity* issues (for instance, Om refers to a configuration where order, but not multiplicity, is the relevant issue). Then, for the reconfigurations events, take a set of modal symbols $\Lambda = \{goto\_OM, goto\_Om, goto\_oM, goto\_om\}$. Consider now the local interfaces. For (S^{om}, F^{om}) choose the usual signature for $Sets$ comprising the set of sorts $S^{om} = \{Elem, Store, Bool\}$ and operation symbols $F^{om}{}_{Store} = \{empty\}$, $F^{om}{}_{Elem\times Store\to Bool} = \{is\_in\}$; $F^{om}{}_{Elem\times Store\to Store} = \{insert\}$; and $F^{om}{}_{\underline{\mathrm{ar}}\to s} = \emptyset$ for the other arities. Clearly, $(S^{Om}, F^{Om}) = (S^{om}, F^{om})$. The remaining cases need to deal with multiplicities; therefore signatures have to be enriched with new operations. Hence, (S^{oM}, F^{oM}) can be defined as $S^{oM} = S^{om} \uplus \{Nat\}$ and $F^{oM}{}_{Elem\times Store\to Nat} = \{mult\}$ and $F^{oM}{}_{\underline{\mathrm{ar}}\to s} = F^{om}{}_{\underline{\mathrm{ar}}\to s}$ for the other arities. Again, $(S^{OM}, F^{OM}) = (S^{oM}, F^{oM})$. Therefore, the following "global" partial signature gets defined: $\big(((S, TF, PF), \mathrm{Nom}, \Lambda), \Gamma\big)$ taking $S = \bigcup_{i\in \mathrm{Nom}} S^i = S^{OM}$, $TF = F^{om}$ and $PF_{Elem\times Store\to Nat} = \{mult\}$ and $PF_{\underline{\mathrm{ar}}\to s} = \emptyset$ for the other arities. On its turn, Γ is defined by the sentences

$@_i(\forall s)(\forall e)\,\mathsf{df}(mult(e, s))$, for $i \in \{oM, OM\}$

$\neg @_i(\forall s)(\forall e)\,\mathsf{df}(mult(e, s))$, for $i \in \{om, Om\}$.

In this setting, we may now proceed with the specification of the global properties, as for example,

$$(\forall e : elem)\, is\_in(e, empty) \overset{\mathrm{s}}{=} False$$

For the local properties one resorts to the hybrid satisfaction operator. This allows, for example, to record the fact that ordering and the multiple insertion are irrelevant for the configuration om:

$$@_{om}(\forall e, e')(\forall s) insert(e', insert(e, s)) \stackrel{s}{=} insert(e, insert(e', s))$$

$$@_{om}(\forall e)(\forall s)\, insert(e, insert(e, s)) \stackrel{s}{=} insert(e, s)$$

On the other hand, the specification of $mult$ in configuration oM is introduced as

$$@_{oM}(\forall e, e')(\forall s) \neg e \stackrel{s}{=} e' \Rightarrow mult(e, insert(e', s)) \stackrel{s}{=} mult(e, s)$$

$$@_{oM}(\forall e) mult(e, empty) \stackrel{s}{=} 0.$$

Finally, we have to specify the possible reconfigurations. For this, one may use sentences as direct as

$$@_{om}\langle goto\_OM\rangle OM$$

stating that a reconfiguration from om to OM is possible, or assuming more elaborated forms as *e.g.*,

$$(\forall e, e')(\forall s)\, insert(e', insert(e, s)) \stackrel{s}{=} insert(e, insert(e', s)) \Rightarrow \langle goto\_Om\rangle Om.$$

The latter states the system can evolve to configuration Om (through the event $goto\_Om$), from any other configuration where the order of insertion is irrelevant.

We conclude here the illustration of the specification method with this (small) small fragment of a buffering component. Notice, however, that several details were not considered here; for example, a definition of the natural numbers and the booleans should be included (and all signatures extended accordingly).

IV. Tool support

In order to have effective practical application, a formal method should have some form of tool support. The specification method discussed in the previous section is no exception. First of all a first-order encoding for $\mathcal{H}EQ$ into FOL was developed. Moreover, through an instantiation of the general method of [8], [4], a first-order encoding for $\mathcal{H}PA$ (or more rigorously, for a version of $\mathcal{H}PA$ with additional contraints regarding the rigidification of operations and sorts [4]) is also available. Both results are instrumental to provide access to effective tool support because FOL enjoys of a very stable set of proof tools.

In particular, integration with the HETS [10] platform seems promising. Using a metaphor of [10], HETS may be seen as a "motherboard" where different "expansion cards" can be plugged in. These pieces are individual logics (with their particular analyzers and proof tools) as well as logic translations, suitably encoded at an institutional level. Details are given in [7], [8].

Notice that HETS already integrates parsers, static analyzers and provers for a wide set of individual logics. Moreover it offers a powerful manager for heterogeneous proofs resorting to the so-called graphs of logics, i.e., graphs whose nodes are logics and, whose edges, are comorphisms between them.

Recently an hybrid version of CASL, called HCASL, as well as the suitable encoding into CASL, were integrated directly in HETS [11]. This provides effective tool support to specifications in the logics discussed in this paper since both are subsumed in HCASL and HETS offers a very stable and mature tool support for the specification in CASL. Other features of HETS can also be explore in this context. For example, the model finder of Darwin, which is already integrated in HETS, may be used as an effective consistency check for our specifications. Moreover, available encodings of FOL into HasCASL, a specification language for functional programs, opens further perspectives for prototyping specifications (see [10]).

V. Conclusions

The paper extends a specification method for reconfigurable systems in order to accommodate the presence of different interfaces (*i.e.*, algebraic signatures) in different configuration states. The extension proposed was smooth and compatible with HETS-based tool support.

Whether similar results can be obtained if other hybrid languages are used, *e.g.*, replacing hybrid equational logic by hybrid first-order logic, remains object of current research.

Acknowledgements: This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT -Portuguese Foundation for Science and Technology within project FCOMP-01-0124-FEDER-028923, *CIDMA-Centro de Investigação e Desenvolvimento em Matemática e Aplicações* of Universidade de Aveiro within the project FCOMP-01-0124-FEDER-022690, and doctoral grant `SFRH/BDE/33650/2009` supported by FCT and *Critical Software S.A., Portugal*.

References

[1] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of IGPL*, 8(3):339–365, 2000.

[2] E. Börger and R. Stärk. *Abstract state machines: A method for high-level system design and analysis*. Springer-Verlag, 2003.

[3] R. Diaconescu. *Institution-independent Model Theory*. Birkhäuser Basel, 2008.

[4] R. Diaconescu and A. Madeira. Encoding hybridized institutions into first order logic. Submited to a journal.

[5] Y. Gurevich. Evolving algebras. In *IFIP Congress (1)*, pages 423–427, 1994.

[6] K. L. J. S. Luca Aceto, Anna Ingólfsdóttir. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

[7] A. Madeira, J. M. Faria, M. A. Martins, and L. S. Barbosa. Hybrid specification of reactive systems: An institutional approach. In G. Barthe, A. Pardo, and G. Schneider, editors, *SEFM*, volume 7041 of *Lecture Notes in Computer Science*, pages 269–285. Springer, 2011.

[8] M. A. Martins, A. Madeira, R. Diaconescu, and L. S. Barbosa. Hybridization of institutions. In A. Corradini, B. Klin, and C. Cîrstea, editors, *CALCO*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

[9] T. Mossakowski, A. Haxthausen, D. Sannella, and A. Tarlecki. CASL: The common algebraic specification language: Semantics and proof theory. *Computing and Informatics*, 22:285–321, 2003.

[10] T. Mossakowski, C. Maeder, M. Codescu, and D. Lucke. Hets user guide - version 0.98. Technical report, DFKI Lab Bremen, 2013.

[11] R. Neves, A. Madeira, M. A. Martins, and L. S. Barbosa. Hybridisation at work. Technical report, INESC-TEC, 2013.